Trabalho Pratico 3
Algoritmos e Estrutura de Dados 3

Lucas Augusto Azevedo

ICEI – Pontifícia Universidade Católica de Minas Gerais (PUC)

Belo Horizonte – MG – Brasil

Resumo. Este documento descreve como foi possível a montagem de um sistema em terminal, simulando o funcionamento de algumas características que um banco faz. Capaz de adicionar clientes e fazer operações CRUDs no mesmo. O intuito do trabalho ˜ foi o entendimento de memória secundário aria, arquivos de dados estruturados ´ e arquivo sequencial e indexado. Além de implementar uma lista invertida, uma busca binaria para substituir as buscas sequenciais e uma ordenação interna no arquivo de índices, para sempre manter a ordenação. Tendo também que ser implementado códigos relacionados a compressão e criptografia. Assim, ocorreu a implementação de um CRUD em arquivo sequencial desenvolvido no TP1, e o desenvolvimento do arquivo de dados e da lista invertida no TP2, o TP3 foi incluindo uma compressão e criptografia nos arquivos de dados.

1. Introdução

Primeiramente, a diferença entre um arquivo sequencial e um indexado é que no sequencial as informações são armazenadas em ordem de inserção, logo, ao realizar uma busca em um arquivo sequencial observamos uma busca de registro por registro, em um arquivo indexado os registros se encontram em um arquivo diferente do arquivo onde a pesquisa e realizada, um arquivo indexado possui um segundo arquivo auxiliar onde são armazenados identificadores e endereços, tornando a pesquisa muito mais rápida, além de possibilitar uma busca não linear, no caso foi implementada a pesquisa binária que é 2X mais rápida.

A criptografia é utilizada desde tempos antigos, e foi muito utilizada nas grandes guerras, quando foram criadas algumas maquinas para ser repassado mensagens, hoje em dia com o grande uso da internet a criptografia é utilizada principalmente para “esconder” mensagens enviadas, e só descriptografando quando o usuário for visualizar. E com a interação na internet sendo muito utilizada veio a necessidade da compressão de dados, para que os mesmos sejam transferidos pela internet de maneira mais eficiente.

2. Desenvolvimento

O trabalho prático 2, foi uma extensão do TP01, mas a proposta do TP02 era ˜ utilizar um sistema de arquivos de índices, e nele fazer a pesquisa do registro desejado através da pesquisa binária, mas para essa pesquisa ser executada os arquivos tem que estar ordenados, dessa forma foi implementado a ordenação externa. O último tópico proposto foi a implementação da lista invertida, na qual foi implementada para fazer a busca por String do nome do usuário, e quando for pesquisar pelo ID, ele utiliza o arquivo de índice.

A proposta do trabalho pratico 3 era inserir a criptografia e a compressão no arquivo de dados, dessa forma foi implementado a criptografia de uma cifra de substituição através de chave e a compressão foi utilizando o algoritmo HUFFMAN.

Dessa forma a classe chamada Menu é a classe main do programa, a classe Crud contém os métodos para o CRUD, foi criado a classe criptografia presente os métodos para criar a cifra, e a classe compressão com os métodos referentes ao algoritmo Huffman. As demais classes foram classe para apoiar as diversas partes do programa, como a classe Sort que auxilia as operações no arquivo de índice, e mais outras classes implementadas:

- ArvoreBMais
- Conta
- Criptografia
- Crud
- HashEstendido
- HuffmanTree
- ListaInvertida
- Menu
- Sort

2.1. Interação com usuário

Inicialmente, o programa já cria a base de dados, com o nome “banco.bin”, além de pegar o ultimo ID inserido no arquivo para que não seja necessária uma navegação geral do arquivo. O programa então, mostra para o usuário um menu, composto pelas seguintes ´ opções ”1 - Criar uma conta bancária”, ”2 - realizar uma transferência”, ”3 - Ler um registro (id)”, ”4 - Atualizar um registro”, ”5 - Deletar um registro (id)”, ”6 - Intercalação,”7 - Lista Invertida”, ”8 - Hash Estendido”, ”9 -”, ”10 - Huffman”, ”0 - Sair”.

2.2. Funções CRUD

2.2.1. Create

Ao receber os dados é dado início ao processo de criação do conta bancária, o qual declara um vetor de bytes (ba), abre o arquivo para escrita, passa todas as informações que foram inseridas pelo usuário e armazenadas no objeto Conta para o vetor de bytes por meio do método da classe Conta, ”toByteArray()”, ´ que, usando as classes ”ByteArrayOutputStream” e ”DataOutputStream”, permitem fazer essa escrita em um vetor de bytes, feito isso, o ponteiro e movido para o início do arquivo e o valor do último ID utilizado é alterado, após isso, o ponteiro é movido de volta para o fim do arquivo e o registro é escrito no arquivo seguindo a seguinte ordem: Lapide(char), tamanho do registro(int) e dados(byte[]).

Feito isso, o arquivo é fechado e o nova conta bancária está criado e registrado com sucesso, caso ocorra algum problema durante a criação, grande parte do método está dentro de um bloco “try” e caso algo dê errado cairá no bloco “catch” que interrompera a execução e mostrara uma alerta ao usuário que algo deu errado durante a criação.

2.2.2. Método de Update

O método Update implementado pode ser completo ou incompleto, quando ele ´ for completo ele irá alterar todos os campos do conta inserido pelo próprio usuário, e quando ele for incompleto ele irá alterar somente o campo escolhido pelo, dessa forma quando o método for incompleto ele não ira precisar mudar de posição no registro principal, pois estará mudando sempre dois números bytes, assim quando for feito esse tipo de Update, os arquivos da lista invertida e de índices não serão alterados, porem quando a alteração é completa é feita o Update tanto no arquivo principal, quanto no arquivo de índice mudando somente o valor do long no registro, e também na lista invertida, encontrando o valor antigo e sobrescrevendo ele pelo novo, alterando assim o registro em todos os arquivos de armazenamento.

O registro a ser atualizado é sempre criptografado para fazer a busca, e quando for inserido a senha atualizada ela é criptografada e adicionada no arquivo principal e nos arquivos da lista e índice.

2.2.3. Método de Delete

O delete dos registros e feito pelo método presente na classe Crud chamado delete, através do índice informado, nele é feito o delete dos elementos nos três arquivos de dados de maneira diferente. No arquivo principal, o delete e feito fazendo uma pesquisa do elemento no arquivo de índice, pegando sua posição e marcando a lápide no arquivo principal, também no arquivo de índice é feito a marcação da lápide após a busca. Mas na lista invertida e´ feito a busca pelo nome do elemento que é pego quando vai marcar a lápide no arquivo principal, e quando acha o elemento o método deletarFile procura pela posição do elemento na LI, e quando acha ele não marca a lápide, ele sobrescreve com zero a posição do registro que quer ser apagado, dessa forma quando for fazer a pesquisa na LI pelo nome e caso agora tenha apagado o único registro presente para esse nome, não será impresso nada, pois tem o nome na LI mas não tem nenhuma posição salva nesse nome na lista invertida.

2.2.4. Método de Pesquisa – Read

No arquivo indexado foi utilizado uma busca binaria, porém para que ela seja executada de maneira correta os elementos presentes tem que estar ordenados, dessa forma utiliza-se a distribuição e a ordenação externa para fazer a ordenação desses registros, e a busca binaria é encarregada de identifica-los.

A busca na Lista Invertida e utilizada para procurar pelo nome do usuário e pela cidade através da classe ListaInvertida e dos métodos listaInvertidaNome e listaInvertidaCidade, dessa forma ela faz a leitura sequencialmente de uma String e compara se e essa a String procurada, caso não seja, ela pula 8 bytes X 20, pois cada Nome na lista tem no máximo 20 espaços para guardar endereços com o mesmo nome, quando acha a String ele vai lendo long por long e concatenando em uma string dividida por ponto e vírgula.

3. Criptografia e Descriptografia

A criptografia escolhida foi a Criptografia de Fluxo - One Time Pad, que consiste em através de uma chave aleatória de mesmo tamanho da senha, realiza uma operação soma com cada caractere da senha e dá a chave para encripto grafar a senha.

O campo criptografado é a Senha, dessa maneira é realizado a criptografia na escrita inicial do registro, salvando a senha criptografada no arquivo principal, é realizada a criptografia na Classe Crud nos métodos Criar, quando e necessário atualizar um registro, a criptografia também está presente no método update.

A descriptografia acontece sempre que o metodo read for chamado, assim ele faz a pesquisa e quando vai imprimir ele descriptografa o campo senha e mostra ela com seu verdadeiro “valor”, utilizando a mesma Criptografia de Fluxo - One Time Pad, porém ele pesquisa o caractere criptografa, e após isso pesquisa qual é o caractere da linha que corresponde a esse caractere criptografado, descobrindo seu verdadeiro valor, fazendo o caractere por caractere até conseguir a String descriptografada.

4. Compressão e Descompressão

A compressão e descompressão foram feitas utilizando o algoritmo Huffamn, através da classe HuffmanTree foi implementado o método cifra para realizar a compressão do arquivo de dados principal. Uma árvore binária completa, chamada de árvore de Huffman é construída recursivamente a partir da junção dos dois símbolos de menor probabilidade, que são então somados em símbolos auxiliares e estes símbolos auxiliares recolocados no conjunto de símbolos. O processo termina quando todos os símbolos forem unidos em símbolos auxiliares, formando uma árvore binária. A árvore é então percorrida, atribuindo-se valores binários de 1 ou 0 para cada aresta, e os códigos são gerados a partir desse percurso.

A codificação de Huffman tem desempenho ótimo quando as probabilidades dos símbolos são potências negativas de dois. A codificação gerada tem também a garantia de ser não ambígua, pois nenhum código pode ser o prefixo de outro código.

A descompressão foi implementada na classe HuffmanTree, e ela descomprime e salva no arquivo de dados principal, além de salvar os dados do índice e da lista referente a compressão nos arquivos principal dos mesmos, deixando assim o arquivo descomprimido pronto para ser utilizado os métodos do CRUD.

5. Testes e Resultados

A criptografia foi realizada nos métodos que realizam alguma escrita em algum arquivo de dado, e na entrada do usuário quando se procura algum registro, a descriptografia foi implementada nos métodos de impressão, resultando em uma escrita e pesquisa correta.

A compressão realiza o Huffman corretamente, além de efetuar um cálculo de melhoria da compressão, salvando esses dados em um novo arquivo de dados, permitindo que a descompressão volte para o arquivo principal esses dados e que eles possam voltar a serem manipulados pelos métodos do CRUD.

As demais funções pedidas durante o trabalho como: Criar uma conta bancária, realizar uma transferência, ler um registro (id), atualizar um registro, deletar um registro (id), intercalação, lista invertida, Hash Estendido e arvoreB Mais, foram todas feitas e funcionam perfeitamente.

6. Conclusão

De acordo com o resultado dos testes, percebemos que o programa está funcionando de forma correta, conseguindo cobrir todos os tipos de situações que podem ser causadas por dados inseridos pelo usuário sendo eles validos ou não.

Pode-se concluir com a adição do arquivo de índice, e com a ordenação, a busca fica completamente mais otimizada, pois no arquivo indexado possui uma forma resumida dos bytes do registro, e quando se pesquisa pode-se utilizar métodos como a busca binária melhorando o tempo de resposta caso tenha um arquivo com muitos registros, e para dar suporte a busca binaria foi realizado a ordenação externa, que é um método para ordenar arquivos em disco, através de caminhos.

Pode-se constatar que a implementação da lista invertida é muito importante, pois ela abre a chance de retornar mais de um registro por pesquisa, caso o usuário tenha o mesmo nome, e caso os usuários sejam da mesma cidade, assim podendo visualizar mais registros através de uma única pesquisa.

Pode-se concluir que apesar que com os impactos significativo na performance e no uso de memória da aplicação, a adição das funções da cripto e compre desempenho um papel impor na optimização de espaço em disco e também na segurança e privacidade dos dados, sem comprometer a robustez e expansibilidade.

Pode-se ´ observar então que o programa está em pleno funcionamento com muitas opções para facilitar a interação do usuário.